ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

MÔN HỌC: VI XỬ LÝ - VI ĐIỀU KHIỂN (TN) (CO3010)

LAB 3 - BUTTONS AND SWITCHES

Giảng viên hướng dẫn:	Lê Trọng Nhân Tôn Huỳnh Long
Lớp:	L02
Sinh viên thực hiện:	Võ Thị Ánh Tuyết - 2213826

Thành phố Hồ Chí Minh, Tháng 09 Năm 2025

Contents

1 Exercise 1: Sketch an FSM 2

2 Exercise 2: Proteus Schematic 2

3 Exercise 3: Create STM32 Project 3

4 Exercise 4: Modify Timer Parameters 6

5 Exercise 5: Adding code for button debouncing 6

6 Exercise 6: Adding code for displaying modes 10

7 Exercise 7: Adding code for increasing time duration value for the red LEDs, amber, greenLEDs 16

8 Link 22

1 Exercise 1: Sketch an FSM

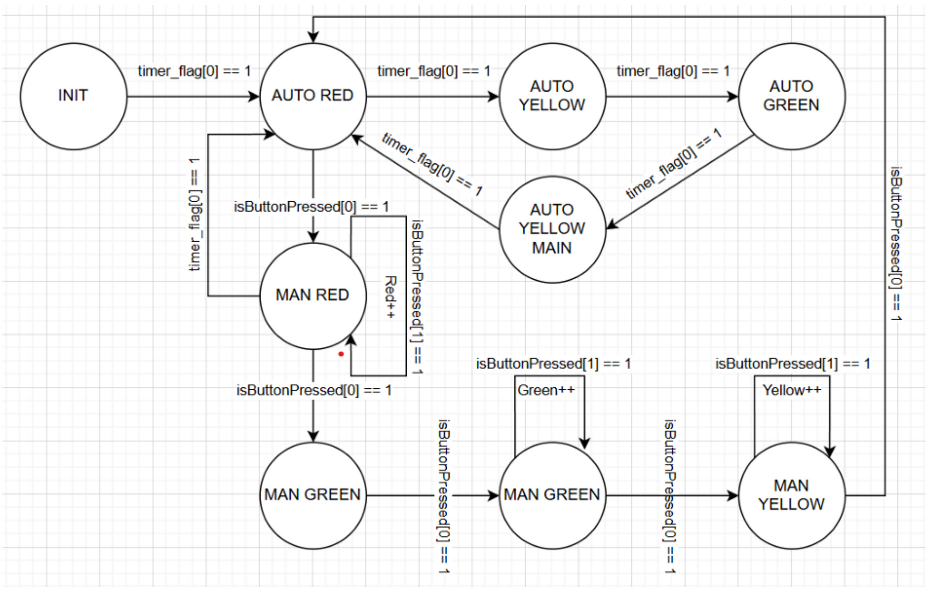

Figure 1: *Exercise 1*

2 Exercise 2: Proteus Schematic

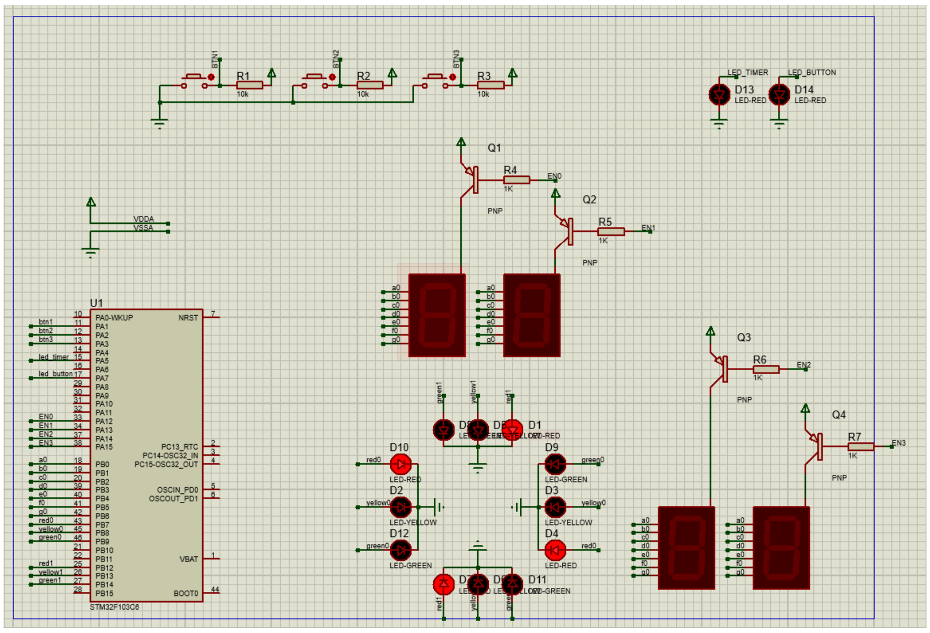

Figure 2: *Exercise 2*

3 Exercise 3: Create STM32 Project

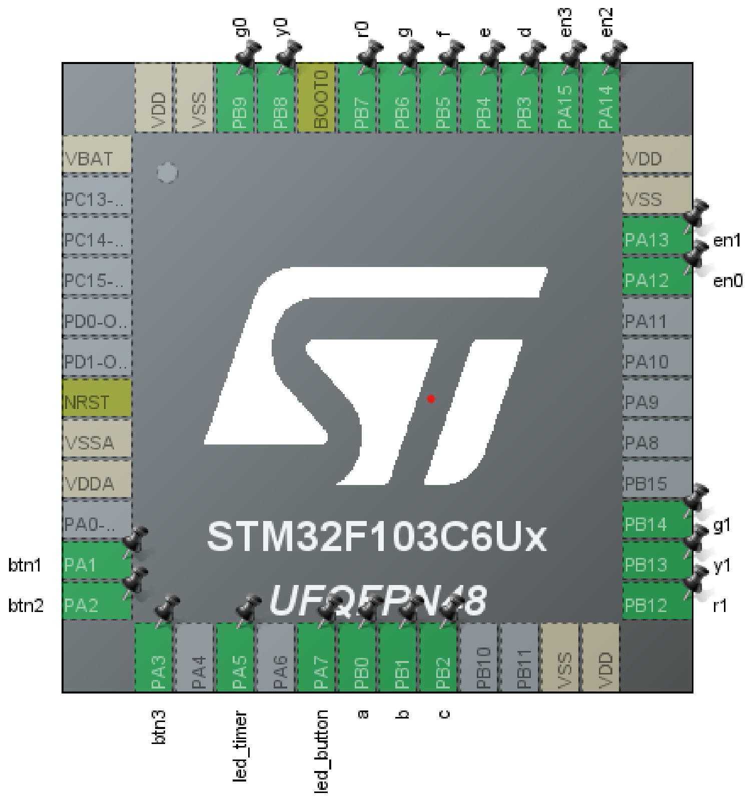

Figure 3: *Exercise 3*

```c
#include "software_timer.h"
int timer_flag[100];
int timer_counter[100];

void set_timer(int index, int counter)
{
  timer_flag[index] = 0;
  timer_counter[index] = counter/tick;
}

void led_timer()
{
  HAL_GPIO_TogglePin(GPIOA, GPIO_Pin_11);
}

void timer_run()
{
  for(int i = 0; i < 10; i++)
  {
    if(timer_counter[i] >= 0)
    {
      timer_counter[i]--;
      if(timer_counter[i] <= 0)
      {
        timer_flag[i] = 1;
      }
    }
  }
}

int is_timer_expired(int index)
{
  if(index < 0 || index >= MAX_TIMERS)
  {
    return -1;
  }
  return timer_flag[index];
}
```

Listing 1: Source code in the software$_t$imer.c

4 Exercise 4: Modify Timer Parameters

I use the variable *tick* to represent the timer interrupt period.
When setting the duration for the software timer, I divide *duration_ms* by *tick* to calculate the number of required interrupts.
Each time the timer interrupt occurs, the function *timer_run()* decreases the counters. When a counter reaches zero, the variable *timer_flag* is set.
With this method, even if the timer interrupt period *(tick = 1 ms, 10 ms, or 100 ms)* changes, the total real-time duration of each timer remains accurate, and the LED continues to blink at exactly *2 Hz*.

5 Exercise 5: Adding code for button debouncing

```c
#include "button.h"
int KeyReg00 = NORMAL_STATE;
int KeyReg10 = NORMAL_STATE;
int KeyReg20 = NORMAL_STATE;
int KeyReg30 = NORMAL_STATE;
int KeyReg01 = NORMAL_STATE;
int KeyReg11 = NORMAL_STATE;
int KeyReg21 = NORMAL_STATE;
int KeyReg31 = NORMAL_STATE;
int KeyReg02 = NORMAL_STATE;
int KeyReg12 = NORMAL_STATE;
int KeyReg22 = NORMAL_STATE;
int KeyReg32 = NORMAL_STATE;

int TimeOutForKeyPress =  50;
int button0_pressed = 0;
int button1_long_pressed = 0;
int button0_flag = 0;
int button1_pressed = 0;
int button1_flag = 0;
int button2_pressed = 0;
int button2_flag = 0;

int isButton0Pressed(){
  if(button0_flag == 1){
    button0_flag = 0;
    return 1;
  }
  return 0;
}
int isButton1Pressed(){
  if(button1_flag == 1){
    button1_flag = 0;
    return 1;
  }
  return 0;
}
int isButton2Pressed(){
  if(button2_flag == 1){
    button2_flag = 0;
    return 1;
  }
  return 0;
}

int isButton1LongPressed(){
  if(button1_long_pressed == 1){
    button1_long_pressed = 0;
    return 1;
  }
  return 0;
}



void getKeyInput()
{
  KeyReg20 = KeyReg10;
  KeyReg10 = KeyReg00;
  // Add your key
  KeyReg00 = HAL_GPIO_ReadPin(btn1_GPIO_Port, btn1_Pin);

  if ((KeyReg10 == KeyReg00) && (KeyReg10 == KeyReg20))
```

```
{
  if (KeyReg20 != KeyReg30)
  {
    KeyReg30 = KeyReg20;
    if (KeyReg30 == PRESSED_STATE)
    {
        TimeOutForKeyPress = 500;
        //HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_7);
        button0_flag = 1;
    }

  }
  else
  {
    TimeOutForKeyPress --;
    if (TimeOutForKeyPress == 0)
    {
      TimeOutForKeyPress = 500;
      if (KeyReg30 == PRESSED_STATE)
      {
        //HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_7);
        button0_flag = 1;
      }
    }
  }
}
KeyReg21 = KeyReg11;
  KeyReg11 = KeyReg01;
  // Add your key
  KeyReg01 = HAL_GPIO_ReadPin(btn2_GPIO_Port, btn2_Pin);

  if ((KeyReg11 == KeyReg01) && (KeyReg11 == KeyReg21))
  {
    if (KeyReg21 != KeyReg31)
    {
      KeyReg31 = KeyReg21;
      if (KeyReg31 == PRESSED_STATE)
      {
        TimeOutForKeyPress = 500;
```

```
          //HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_7);
          button1_flag = 1;
        }
      }
      else
      {
        TimeOutForKeyPress --;
        if (TimeOutForKeyPress == 0)
        {
          TimeOutForKeyPress = 500;
          if (KeyReg31 == PRESSED_STATE)
          {
            //HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_7);
            button1_flag = 1;
          }
        }
      }
    }
    KeyReg22 = KeyReg12;
    KeyReg12 = KeyReg02;
      // Add your key
    KeyReg02 = HAL_GPIO_ReadPin(btn3_GPIO_Port, btn3_Pin);

    if ((KeyReg12 == KeyReg02) && (KeyReg12 == KeyReg22))
    {
      if (KeyReg22 != KeyReg32)
      {
        KeyReg32 = KeyReg22;
        if (KeyReg32 == PRESSED_STATE)
        {
          TimeOutForKeyPress = 500;
          //HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_7);
          button2_flag = 1;
        }
        }
      else
      {
        TimeOutForKeyPress --;
            if (TimeOutForKeyPress == 0)
            {
              TimeOutForKeyPress = 500;
              if (KeyReg32 == PRESSED_STATE)
              {
                //HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_7);
                button2_flag = 1;
              }
            }
      }
    }
}
```

Listing 2: Source code button.c

6 Exercise 6: Adding code for displaying modes

To add code for display mode on seven-segment LEDs

```
#include "fsm_auto.h"

int i = 0;
int led_buffer[4] = {0, 0, 0, 0};
int num_to_display_1 = 0;
int num_to_display_2 = 0;

void fsm_auto()
{
  if(is_timer_expired(3) && (status == AUTO_GREEN_RED ||
                 status == AUTO_RED_GREEN ||
                 status == AUTO_YELLOW_RED ||
                 status == AUTO_RED_YELLOW ||
                 status == AUTO_INIT))
  {
    getKeyInput();
    if (isButton0Pressed())
    {
      HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_7);
      clear();
      led_buffer[0] = 0;
      led_buffer[1] = 0;
      led_buffer[2] = 0;
      led_buffer[3] = 0;
      status = MANUAL_INIT;
    }
    else set_timer(3, 20);
  }

  if(is_timer_expired(5)  && (status == AUTO_GREEN_RED ||
                            status == AUTO_RED_GREEN ||
                    status == AUTO_YELLOW_RED ||
                    status == AUTO_RED_YELLOW ||
                    status == AUTO_INIT))
  {
    num_to_display_1--;
    num_to_display_2--;
    led_buffer[0] = num_to_display_1/10;
    led_buffer[1] = num_to_display_1%10;
    led_buffer[2] = num_to_display_2/10;
    led_buffer[3] = num_to_display_2%10;
    set_timer(5, 1000);
  }
  if(is_timer_expired(4))
  {
    if(i == 4) i = 0;
    update7SEG(i);
    display7SEG(led_buffer[i]);
    i++;
    set_timer(4, 150);
  }

  if(is_timer_expired(2))
  {
    switch (status)
    {
      case AUTO_INIT:
        status = AUTO_RED_GREEN;
        break;
      case AUTO_RED_GREEN:
        led_red_and_green();
```

```c
        num_to_display_1 = 5;
        num_to_display_2 = 3;
        status = AUTO_RED_YELLOW;
        set_timer(2, 3000);
        break;
      case AUTO_RED_YELLOW:
        led_red_and_yellow();
        num_to_display_1 = 2;
        num_to_display_2 = 2;
        status = AUTO_GREEN_RED;
        set_timer(2, 2000);
        break;
      case AUTO_GREEN_RED:
        led_green_and_red();
        num_to_display_1 = 3;
        num_to_display_2 = 5;
        status = AUTO_YELLOW_RED;
        set_timer(2, 3000);
        break;
      case AUTO_YELLOW_RED:
        led_yellow_and_red();
        num_to_display_1 = 2;
        num_to_display_2 = 2;
        status = AUTO_GREEN_RED;
        set_timer(2, 2000);
        break;
      default:
        break;
    }
  }
}
```

Listing 3: Sourcecodeof fsm_auto.c

To add code for blinking LEDs depending on the mode that is selected.

```c
#include "fsm_manual.h"

int red_time = 0;
int yellow_time = 0;
int green_time = 0;
int red;
int yellow;
int green;
int check = 0;
int num_to_display1 = 0;
int num_to_display2 = 0;

void fsm_manual()
{
  if(is_timer_expired(8))
  {
    getKeyInput();
    set_timer(8, 20);
  }

  if(is_timer_expired(9))
  {
    switch(status)
    {
    case MANUAL_INIT:
      toggle();
      led_buffer[0] = 0;
      led_buffer[1] = 0;
      led_buffer[2] = 0;
      led_buffer[3] = 0;
      if(isButton0Pressed())
      {
        clear();
        status = RED_SET;
      }
      else if(isButton2Pressed())
      {
        clear();
        status = AUTO_INIT;
      }
      else set_timer(3, 20);
      set_timer(9, 500);
      break;
    case MAN_RED_GREEN:
      led_red_and_green();
      led_buffer[0] = red/10;
      led_buffer[1] = red%10;
      led_buffer[2] = green/10;
      led_buffer[3] = green%10;
      red--;
      green--;
      if(green == 0)
      {
        green = green_time;
        status = MAN_RED_YELLOW;
      }
      if(isButton2Pressed())
      {
        clear();
        red_time = 0; yellow_time = 0; green_time = 0;
        status = MANUAL_INIT;
        set_timer(3, 20);
      }
      set_timer(9, 1000);
      break;
    case MAN_RED_YELLOW:
      led_red_and_yellow();
      led_buffer[0] = red/10;
      led_buffer[1] = red%10;
      led_buffer[2] = yellow/10;
      led_buffer[3] = yellow%10;
      red--;
      yellow--;
      if(red == 0)red = red_time;
      if(yellow == 0)
      {
        yellow = yellow_time;
        status = MAN_GREEN_RED;
      }
      if(isButton2Pressed())
      {
        clear();
        red_time = 0; yellow_time = 0; green_time = 0;
        status = MANUAL_INIT;
        set_timer(3, 20);
      }
      set_timer(9, 1000);
      break;
    case MAN_GREEN_RED:
      led_green_and_red();
      led_buffer[0] = green/10;
      led_buffer[1] = green%10;
      led_buffer[2] = red/10;
      led_buffer[3] = red%10;
      red--;
      green--;
      if(green == 0)
      {
        green = green_time;
        status = MAN_YELLOW_RED;
      }
      if(isButton2Pressed())
      {
        clear();
        red_time = 0; yellow_time = 0; green_time = 0;
        status = MANUAL_INIT;
        set_timer(3, 20);
      }
      set_timer(9, 1000);
      break;
    case MAN_YELLOW_RED:
      led_yellow_and_red();
      led_buffer[0] = yellow/10;
      led_buffer[1] = yellow%10;
      led_buffer[2] = red/10;
      led_buffer[3] = red%10;
      yellow--;
      red--;
      if(yellow == 0)
      {
        red = red_time;
```

```
          yellow = yellow_time;
          status = MAN_RED_GREEN;
        }
        if(isButton2Pressed())
        {
          clear();
          red_time = 0; yellow_time = 0; green_time = 0;
          status = MANUAL_INIT;
          set_timer(3, 20);
        }
        set_timer(9, 1000);
        break;
      default:
        break;
        }
  }
}
```

Listing 4: Source code of fsm_manual.c

7 Exercise 7: Adding code for increasing time duration value for the red LEDs, amber, greenLEDs

In this exercise, this source code I show is also used for Exercise 8, 9

```
#include "fsm_manual.h"

int red_time = 0;
int yellow_time = 0;
int green_time = 0;
int red;
int yellow;
int green;
int check = 0;

int num_to_display1 = 0;
int num_to_display2 = 0;

void fsm_manual()
{
```

```
if(is_timer_expired(8))
{
  getKeyInput();
  set_timer(8, 20);
}

if(is_timer_expired(9))
{
  switch(status)
  {
  case MANUAL_INIT:
    toggle();
    led_buffer[0] = 0;
    led_buffer[1] = 0;
    led_buffer[2] = 0;
    led_buffer[3] = 0;
    if(isButton0Pressed())
    {
      clear();
      status = RED_SET;
    }
    else if(isButton2Pressed())
    {
      clear();
      status = AUTO_INIT;
    }
    else set_timer(3, 20);
    set_timer(9, 500);
    break;
  case RED_SET:
    togglered();
    if(isButton0Pressed())
    {
      clear();
      if (red_time <= 1) red_time = 2;
      else red = red_time;
      status = YELLOW_SET;
      led_buffer[0] = yellow_time/10;
      led_buffer[1] = yellow_time%10;
```

```
        led_buffer[2] = 0;
        led_buffer[3] = 0;
        set_timer(3, 20);
      }
      else if(isButton1Pressed())
      {
        clear();
        red_time++;
        led_buffer[0] = red_time/10;
        led_buffer[1] = red_time%10;
        led_buffer[2] = 0;
        led_buffer[3] = 0;
        set_timer(3, 20);
      }
      else if(isButton2Pressed())
      {
        clear();
        if(red_time <= 1)
        {
          red_time = 2;
          yellow_time = 1;
          green_time = red_time - yellow_time;

          red = red_time;
          yellow = yellow_time;
          green = green_time;

          status = MAN_RED_GREEN;
        }
        else
        {
          yellow_time = 1;
          green_time = red_time - yellow_time;
          red = red_time;
          yellow = yellow_time;
          green = green_time;
          led_buffer[0] = red_time/10;
          led_buffer[1] = red_time%10;
          led_buffer[2] = 0;
          led_buffer[3] = 0;
          status = MAN_RED_GREEN;
        }
        set_timer(3, 20);
      }
      else set_timer(3, 20);
      set_timer(9, 500);
      break;
    case YELLOW_SET:
      toggleyellow();
      if(isButton0Pressed())
      {
        clear();
        if (yellow_time <= 1) yellow_time = 1;
        status = GREEN_SET;
        led_buffer[0] = green_time/10;
        led_buffer[1] = green_time%10;
        led_buffer[2] = 0;
        led_buffer[3] = 0;
        set_timer(3, 20);
      }
      else if(isButton1Pressed())
      {
        yellow_time++;
        led_buffer[0] = yellow_time/10;
        led_buffer[1] = yellow_time%10;
        led_buffer[2] = 0;
        led_buffer[3] = 0;
        set_timer(3, 20);
      }
      else if(isButton2Pressed())
      {
        clear();
        if(yellow_time < 1)
        {
          yellow_time = 1;
          green_time = red_time - yellow_time;
          red = red_time;
          yellow = yellow_time;
          green = green_time;
          led_buffer[0] = yellow_time/10;
          led_buffer[1] = yellow_time%10;
          led_buffer[2] = 0;
          led_buffer[3] = 0;
          status = MAN_RED_GREEN;
        }
        else if (yellow_time >= red_time)
        {
          red_time = 0;
          yellow_time = 0;
          green_time = 0;
          led_buffer[0] = 0;
          led_buffer[1] = 0;
          led_buffer[2] = 0;
          led_buffer[3] = 0;
          status = RED_SET;
        }
        else{
          yellow = yellow_time;
          green = green_time = red_time - yellow_time;
          status = MAN_RED_GREEN;
        }
        set_timer(3, 20);
      }
      else set_timer(3, 20);
      set_timer(9, 500);
      break;
    case GREEN_SET:
      togglegreen();
      if(isButton0Pressed())
      {
        clear();
        status = RED_SET;
        set_timer(3, 20);
      }
      else if(isButton1Pressed())
      {
        green_time++;
        led_buffer[0] = green_time/10;
        led_buffer[1] = green_time%10;
        led_buffer[2] = 0;
        led_buffer[3] = 0;
        set_timer(3, 20);
      }
      else if(isButton2Pressed())
      {
        if(green_time == (red_time - yellow_time) && (yellow_time
   != 0))
        {
          clear();
          status = MAN_RED_GREEN;
        }
        else {
          status = RED_SET;
          red_time = 0;
          yellow_time = 0;
          green_time = 0;
          led_buffer[0] = 0;
          led_buffer[1] = 0;
          led_buffer[2] = 0;
          led_buffer[3] = 0;
        }
        set_timer(3, 20);
      }
      else set_timer(3, 20);
      set_timer(9, 500);
      break;
    default:
      break;
      }
  }

}
```

Listing 5: Source code of fsm_manual.c

8 Link

For more details. please refer to this source code **Link Github**